U. A. M. IZTAPALAPA BIBLIOTECA

UNIVERSIDAD AUTONOMA METROPOLITANA

UNIDAD IZTAPALAPA

C. B. I.

REPORTE FINAL DE:

PROYECTO DE INGENIERIA ELECTRONICA II

CHECADOR DE TARJETAS

146498

ALUMNOS:

ACOSTA GASTELUM, LUZ FRANCISCA.

PULIDO BAÑOS ALEJANDRO.

ASESOR:

SERGIO PAEZ RODEA.

MAYO DE 1990

U. A. M. IZTAPALAPA BIBLIOTECA

INDICE

PROLOGO	3
GENERALIDADES	4
COMUNICACION ASINCRONA	4
DIAGRAMA A BLOQUES	5
PUERTO SERIE	6
DESARROLLO	9
RELOJ MAESTRO	11
TABLA DE ESTADOS	13
ANALISIS DE RESULTADOS	19
CONCLUSIONES	22
BIBLIOGRAFIA	23
APENDICE A: INFORMACION UARTS	24
APENDICE B: INFORMACION REGISTROS 82050	34
APENDICE C: DIAGRAMA DE LA INTERFASE	36
APENDICE D: PROGRAMA DEL PROYECTO	38

U. A. M. IZTAPALAPA BIBLIOTECA

P R O L O G O

La función principal de la sección de Computadoras, perteneciente al sistema de tranporte colectivo METRO, y para la cual ha sido realizado este proyecto, es la del mantenimiento de las computadoras y equipo periférico que existe para cada una de las lineas del metro.

Esas computadoras tienen encargada la tarea de regular el tránsito de trenes, llevar el seguimiento de éstos a lo largo de toda la linea, y dar el despacho de trenes en las terminales.

La comunicación entre la sección donde se maneja el tráfico de trenes (Teletráfico) y la computadora correspondiente a cada linea se realiza a través de una interfase, la cual hace uso de diferentes tarjetas para la transmisión y recepción de datos; entre estas una de las mas usadas es la tarjeta de entradas digitales opto-acopladas de 16 bits RTP 7435/37, cuya función es la de recibir las entradas digitales del sistema de teletráfico procesarlas y pasarlos a la tarjeta RTP 7435/38 salida a relés de 16 bits, para que de ahí sean enviados a la computadora.

El propósito de este proyecto fué el de elaborar un probador para la tarjeta de entradas digitales opto-acopladas para que con esto la reparación de dicha tarjeta sea de una forma mas rápida, de forma tal que se pod[illegible] agilizar el sistema de mantenimiento y ademas que se ahorrará dinero reparando las tarjetas y no sustituyéndolas por nuevas.

G E N E R A L I D A D E S

Para la realización de nuestro proyecto, se requirió diseñar una interfase que comunicara la PC con la tarjeta a probar y viceversa; así como de un programa que mandara y recibiera datos por el puerto serie de la PC utilizando comunicación asíncrona y la norma RS-232 del puerto serie en la computadora.

La tarea principal de la interfase es la de recibir dos bytes de prueba que son mandados por la PC a través del puerto serie; una vez recibidos mandarlos a las entradas en paralelo de la tarjeta a probar y generar un retardo para posteriormente recibir también en paralelo los datos de salida de la tarjeta; estos son recibidos en dos bytes, los cuales son mandados en forma serie al puerto de la PC. Un diagrama a bloques de la interfase se muestra en la figura 1.

Para lograr lo anterior se diseñó una parte de control, pero se requirió de unos circuitos especiales capaces de transmitir y recibir de forma asíncrona, estos son los UARTS AY-5-1013A (información en el apéndice A).

COMUNICACION ASINCRONA:

La transmisión de información digital a través de un par de alambres no es del todo difícil. De hecho, un registro sencillo que convierta de serie a paralelo tal como el SN 74165 puede ser usado como un transmisor y un registro convertidor de serie a paralelo, tal como el SN 74164 que puede ser usado como receptor. Pero todo este proceso solo contiene la carga de datos en paralelo dentro del transmisor y de su carga en el registro, un bit a la vez. El receptor puede mandar los datos recibidos en el registro uno a la vez hasta que sean todos recibidos. Desafortunadamente, este proceso no es una transmisión asíncrona, dado que las señales del reloj son comunes y muchas señales de control son requeridas entre los dos registros de carga integrados en los circuitos. Podría sin embargo trabajar, pero no va a ser compatible con el estandar de las tecnicas de transferencia en serie, de datos en forma asíncrona. En una verdadera transferencia de datos, solo la linea de transmisión de datos se conecta al receptor y al transmisor.

DIAGRAMA A BLOQUES DE LA INTERFASE

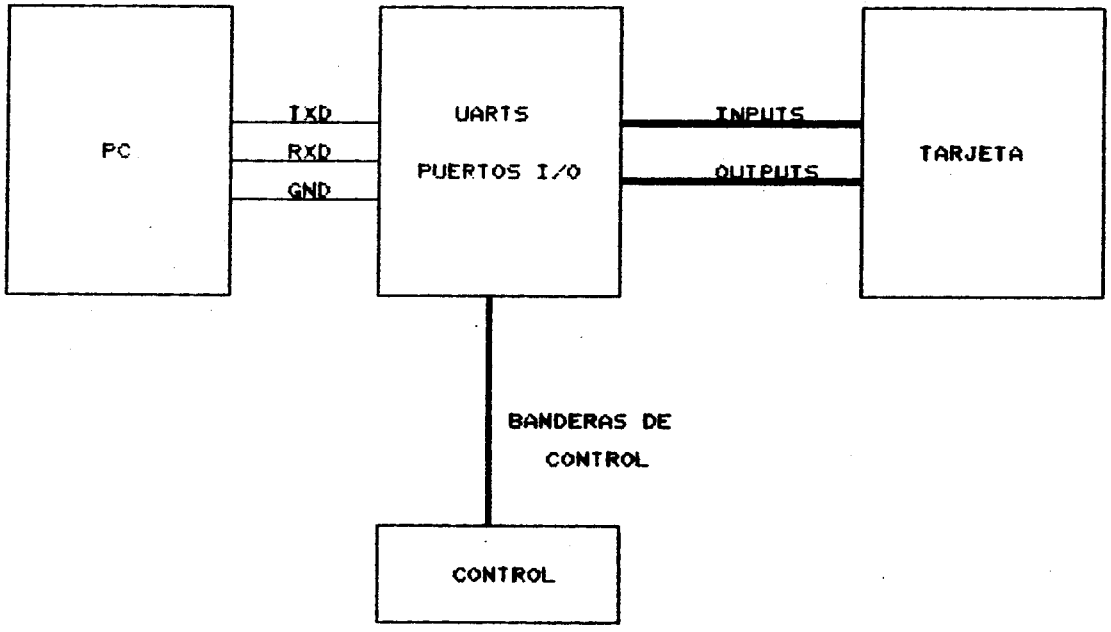


Figura 1

Obviamente, debe de haber algunos trucos que son usados para controlar una verdadera transferencias de datos en serie, de forma asíncrona.

Dado que en una comunicación asíncrona no existe un reloj común conectado entre el transmisor y el receptor, y tambien ninguna señal de control existe entre ellos, un BIT DE COMIENZO siempre precede a cualquier transferencia de datos. Este bit es siempre un cero lógico. Este es usado para indicar a el receptor que una nueva serie de datos ya ha sido transmitida.

Cada transmisión termina con uno o dosBITS DE PARADA, los cuales son siempre uno lógicos. La opción de uno o dos BITS DE PARADA son de alguna manera arbitraria. En algunos casos especiales la aplicación de uno, dos bits o uno y medio bits de parada pueden ser usados.

Por supuesto que debe de haber un límite en cuanto al número de bits de datos en cada transmisión, por otro lado algunos sistemas transmiten cinco bits de datos despues de un bit de comienzo, mientras otros desearían transmitir cientos de bits de datos. La serie de bits de datos encontrados entre el bit de comienzo y el bit de parada esta limitada entre cinco y ochobits.

El BIT DE PARIDAD, par o no, es usado para checar los errores en la transmisión de datos. El BIT DE PARIDAD indica si el número de unos lógicos en la palabra de datos es par o impar. La figura 2 muestra los bits de control para la comunicación asíncrona.

PUERTO SERIE (NORMA RS 232)

El corazón de el adaptador es un chip INS8250 o un equivalente. Algunas caracteristicas son las siguientes: No hay necesidad de una sincronización precisa; el reloj del receptor es independiente; detección de un bit de comienzo falso; generación de una linea de aborto y detección.

Todo el protocolo de comunicación es una función del sistema y debe de ser cargada antes de que el adaptador esté en operación. Todos los Status de las señales de control deben de ser manejados a través del sistema de Software. La figura 3 es un diagrama a bloques del adaptador de comunicación asíncrona.

FORMATO DE DATOS

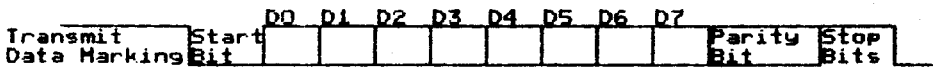


Figura 2

DIAGRAMA A BLOQUES DEL ADAPTADOR DE COMUNICACION ASINCRONA

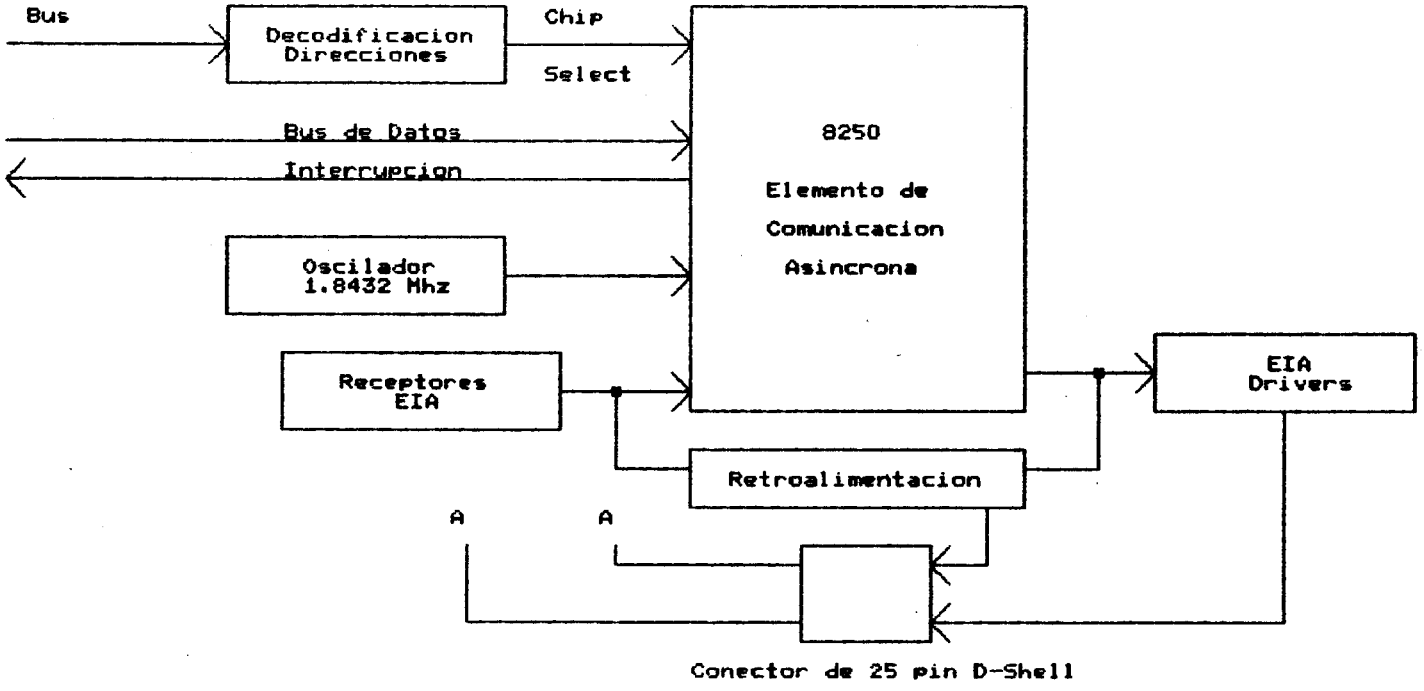


Figura 3

Los diferentes modos de operación son seleccionados programando los elementos de comunicación asíncrona del 8250. Esto es hecho a través de seleccionar las direcciones de entrada y salida (primaria hex 3f8 a 3ff, y hex 2f8 a 2ff para la secundaria) escribiendo el dato de salida en el conector. Los bits de direccionamiento A0, A1, y A2 seleccionan los diferentes registros que definen los diferentes modos de operación. También, el bit de acceso del Latch divisor (bit 7) del registro de linea de control es usado para seleccionar ciertos registros.

En la tabla 1 se muestran las direcciones correspondientes a los registros utilizados para manejar el 8250.

DIRECCION DE E/S ADAPTADOR PRIMARIO	REGISTRO SELECCIONADO
3F8	Buffer TX
3F8	Buffer RX
3F8	Latch Divisor LSB
3F9	Latch Divisor MSB
3F9	Registro de Interrupt Enable
3FA	Registro de Identificación de Interrupción
3FB	Registro de Control de Linea
3FC	Registro de Control de Modem
3FD	Registro de Status de Linea
3FE	Registro de Status de Modem

TABLA 1

La tabla del apéndice B muestra el mapa de registros donde se encuentra la descripción de los bits para cada registro mencionado en la tabla 1.

D E S A R R O L L O

Para la utilización de los UARTS en la interfase; la parte de control, recepción y transmisión que comprenden el modo de funcionamiento de este tipo de circuitos, quedó de la siguiente manera:

En la parte de control se programó el UART para que no trabajara con paridad, tuviera un bit de parada y ocho bits de datos; quedando conectada esta parte como se muestra en la tabla 2a.

Para la parte de transmisión se recibían ocho bits en paralelo y se usó un reloj 16 veces mayor al BAUD RATE generado por la PC; además de utilizar sus banderas para conocer el estado del buffer de transmisión, la carga de datos y el final de la transmisión, la cual por supuesto es en forma serie. La tabla 2b muestra como fué utilizada esta parte.

En la parte de recepción tenemos ocho salidas en paralelo, una entrada en serie, otra vez un reloj 16 veces el BAUD RATE generado por la PC y la parte que programa el protocolo de recepción de los datos. Este protocolo utiliza las banderas del framing error y la de dato recibido disponible. En la tabla 2c se encuentra la forma en que esta parte fue conectada.

Una vez expuesto lo anterior y dado que estamos utilizando una comunicación asíncrona es de suponer que la transmición y recepción en la PC haya sido programada de la misma forma.

NUM. DEL PIN	NOMBRE	VALOR
21	MR1	lógico al inicio y 0 durante el resto del tiempo.
34	CS	1 lógico
35	NP	0 lógico
36	SB	0 lógico
37	NB-2	1 lógico
38	NB-1	1 lógico
16	SWE	0 lógico

TABLA 2a

NUM. DEL PIN	NOMBRE	VALOR
26 - 33	DB1 - DB8	Salidas de la tarjeta
40	CLOCK	Control del reloj
25	SO	Puerto serie de la PC
22	TBMT	Control de la interfase
23	DS	Control de la interfase
24	EOC	Control de la interfase

TABLA 2b

NUM. DEL PIN	NOMBRE	VALOR
12 - 5	RD1 - RD8	Entradas de la tarjeta
17	CLOCK	Control de reloj
20	SI	Puerto serie de la PC
19	RDA	Control de la interfase
14	FE	Control de la interfase

TABLA 2c

RELOJ MAESTRO

El reloj maestro genera frecuencias desde 2400 hasta 153600 Hz. con objeto de poder recibir datos desde 150 hasta 9600 baudes. Esto se realizó por medio de un reloj oscilando a 153600 Hz.

Como la transmisión es asíncrona, la frecuencia del reloj maestro no necesita ser muy exacta, con que no varie mas del tres porciento de la frecuencia dada se asegura que se este ejecutando la recepción y la transmisión. Para la realización del reloj se utilizaron un temporizador LM555 como generador de la frecuencia, y un 14040 como divisor de frecuencias. En la figura 4 se muestra el circuito del reloj maestro.

En la tabla 3 mostramos las frecuencias con sus bauds.

FRECUENCIA	BAUDES
2400	150
4800	300
9600	600
19200	1200
38400	2400
76800	4800
153600	9600

TABLA 3

RELOJ MAESTRO

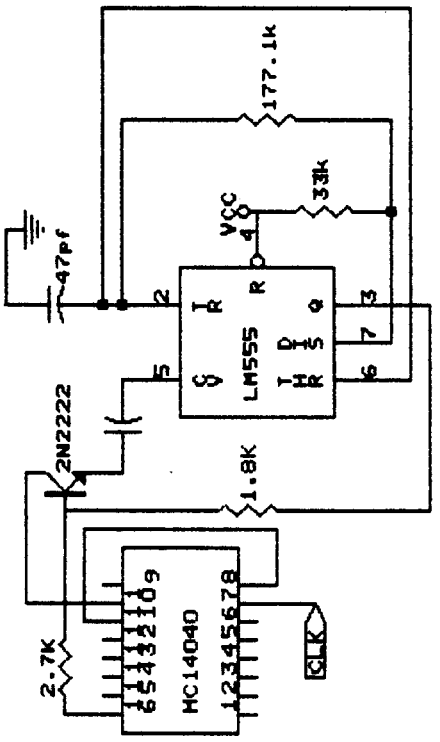


Figura 4

Basandonos en los estados en los que deben de estar las banderas de los UARTS dependiendo de la función que este realizando la interfase se propuso una tabla de estados la cual se muestra en la tabla 4; esta tabla de estados es la que nos sirvio para hacer la parte de control de la interfase.

TABLA DE ESTADOS

bandera	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
RDA1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TMBT1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1
EOC1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
CLKT1	0	0	0	0	0	0	0	0	0	0	0	0	CLOCK		DE	T1
CLKR1	0	CLK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RDA2	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
TMBT2	1	1	1	1	1	0	0	1	1	0	0	1	0	0	0	0
EOC2	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1
CLKT2	0	0	0	0	CLOCKT2		0		CLOCKT2			0	0	0	0	0
CLKR2	0	0	CLK	0	0	0	0	0	0	0	0	0	0	0	0	0
A'	1	1	1	0	0	1	1	0	0	1	1	0	1	1	1	1
MR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
VAR1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
VAR2	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0
DS2	0	0	0	0	/	0	0	0	/	0	0	0	0	0	0	0
DS1	0	0	0	0	0	0	0	/	0	0	0	/	/	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Q'	1	1	1	/	1	1	1	/	1	1	1	/	1	1	1	1

Nota: El simbolo / en esta tabla viene representando un pulso megativo (es decir un pulso que va de un nivel 1 a 0 y regresa a un nivel 1).

TABLA 4

La tabla 4 antes presentada ya contiene algunos estados que resultarón despues del manejo de las variables artificiales, las cuales son M, VAR1, VAR2; de estas junto con las banderas desplegadas por los UARTS dependen las variables DS1, DS2, CLKT1, CLKT2, CLKR1, CLKR2 y el MR.

Las ecuaciones de la parte de control quedaron de la siguiente manera.

```
CLKR1 = CLOCK * RDA1' * MR'
CLKR2 = CLOCK * RDA1 *RDA2'
CLKT1 = CLOCK * M
CLKT2 = (CLOCK * M') * RDA2
   A' = (RDA2' + TMBT2') * EOC2 + EOC2'
  DS2 = VAR1 * Q2' (74123)
  DS1 = VAR2 * Q2' (74123)
```

Los mapas para las variables CLKR1 y CLKR2 son los siguientes:

MR	RDA1	RDA2 \ TMBT1 TMBT2	00	01	11	10
0	0	0	X	X	1	X
0	0	1	X	X	X	X
0	1	1	0	X	0	0
0	1	0	X	X	0	X
1	1	0	X	X	X	X
1	1	1	X	X	X	X
1	0	1	X	X	X	X
1	0	0	X	X	0	X

CLKR1=RDA1'*MR'*CLK

RDA1	TMBT1 \ RDA2 TMBT2	00	01	11	10
0	0	X	X	X	X
0	1	X	0	X	X
1	1	X	1	0	0
1	0	X	X	X	0

CLKR2=RDA1*RDA2'*CLK

146498

Las variables VAR1, VAR2, M y MR se implementarón con circuitos 7474 que son flip-flops tipo D con dos en cada integrado.

La forma en que fueron conectados los flip-flops para la var1 fue de la siguiente manera:

CLEAR = RDA2, D = RDA2, CLK = Q1' (74123), PRESET = 1, SALIDA Q a la entrada de una and, cuya otra entrada esta conectada a TMBT2, y a la salida tenemos VAR1. Aqui se deja pasar un uno en el flip-flop al darse un flanco positivo en el reloj el cual esta conectado al retardo generado por el 74123 (figura 5).

La forma para VAR2 es la siguiente:

CLEAR = RDA2 * TMBT1, D = 1, CLK = TMBT2, PRESET = 1, y en la salida Q tenemos VAR2. El cambio de estado a la salida se debe a un flanco positivo generado por la señal TMBT2 la cual nos indica que el buffer de transmisión vuelve a estar listo (fig. 6).

La forma para M es la siguiente, pero en esta ocasion se usaron los dos flip-flops del integrado:

CLEAR1 = RDA2 = CLEAR2, D1 = 1 = D2, CLK1 = BIT 1 de un contador binario, PRESET1 = 1 =PRESET2, Q1 = CLK2, Q2 es M y la salida Q2' es M'. En esta ocasión el nivel de la salida es cambiado debido a un flanco positivo generado por el bit 2 de un contador permitiendonos así manadar dos veces un byte a través del UART2 (fig. 7).

Para Master Reset fue necesario separarlo en dos debido a que necesitamos un reset para la interfase para cuando esta empieza a trabajar, este consiste en un diodo, una resistencia y un capacitor, su configuración puede ser encontrada en el diagrama de la figura 8 . La segunda parte que compone al reset tiene como función la de dar un reset cuando hemos terminado un ciclo de estados. La segunda parte fue implementada con un flip-flop debido a que este tipo de circuitos con un solo cambio de nivel detectado es posible hacerlo cambiar de estado a la salida. Así la segunda parte quedo implementada de la siguiente manera:

Tambien esta ocasión fueron utilizados los dos flip-flops contenidos en el integrado. CLEAR1 = CLEAR2 = RDA2, D1 = D2 =1, CLK1 =EOC1, PRESET1 =PRESET2 =1, Q1 =CLK2, y por último la salida Q2 proporciona la señal para el reset. El flanco positivo esta vez es generado por la señal EOC1 la cual nos indica con ello que se ha dejado de transmitir a través del UART1 y por lo tanto se han terminado de mandar bytes de la interfase a la PC

Los Data Strobe tanto del Uart1 como del Uart2 dependen de un pulso generado por la segunda parte contenida en el integrado 74123, este pulso es dejado pasar o no a la respectiva pata del UART por VAR1 o VAR2 según corresponda. Un pulso en la pata de Data Strobe del UART genera que se carge un dato y se comienze su transmisión. (Para mas información sobre el funcionamiento de los pines del UART consultar el apendice A al final del reporte).

Debido a que la interfase trabaja a una velocidad mas rapida a la de la respuesta de la tarjeta, hubo la necesidad de generar un retardo para controlar el envio de los bytes de respuesta a la PC, este fue implementado con un circuito 74123 el cual utiliza un capacitor de 100 pF y una resistencia de 35 K ohms lo cul nos da un tw de 1000 ns, las especificaciones sobre este integrado se pueden encontrar en el apendice C. Este circuito de retardo puede ser encontrado en la figura 9.

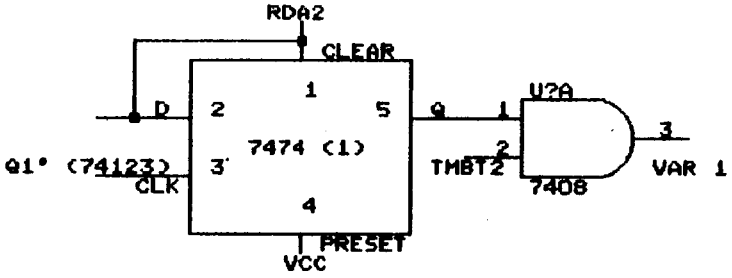


Figura 5

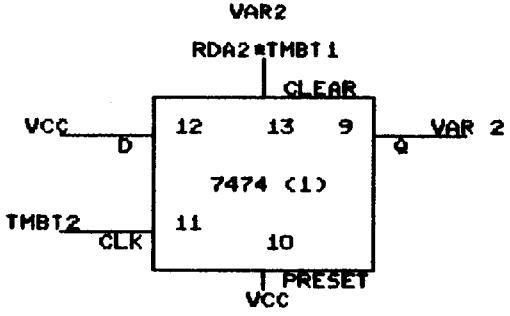


Figura 6

VARIABLE M

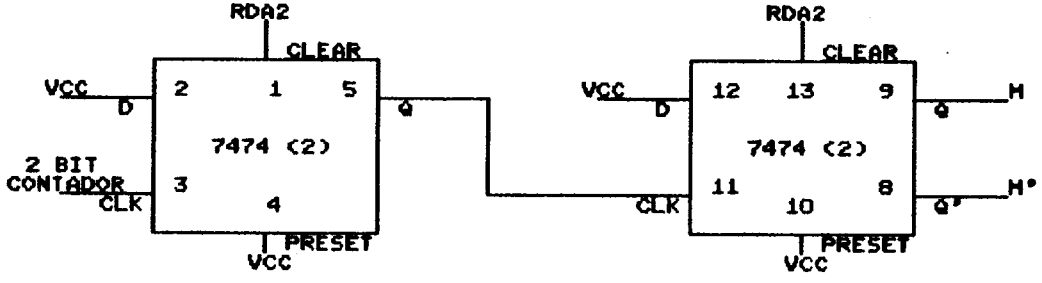


Figura 7

MASTER RESET

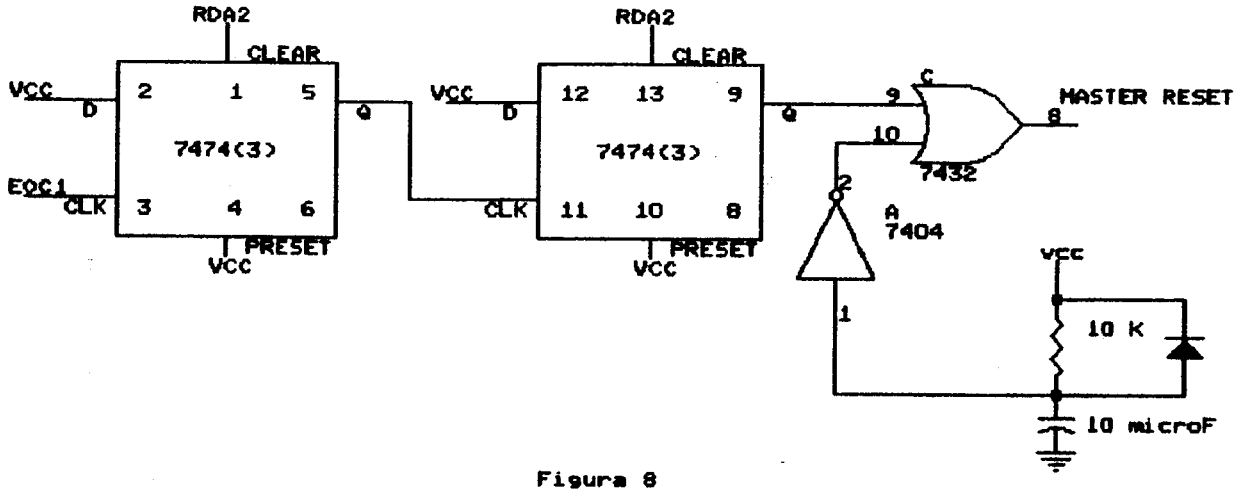


Figura 8

CIRCUITO DE RETARDO

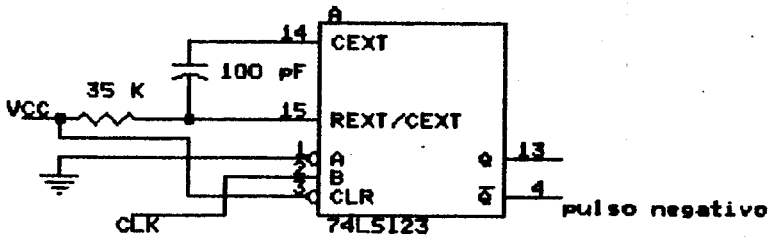


Figura 9

ANALISIS DE RESULTADOS

Al realizar pruebas por separado, tanto a la interfase como al programa ambos hechos por nosotros, pudimos comprobar que los dos trabajan satisfactoriamente. Sin embargo, al ensamblar y hechar todo a andar localizamos ciertos detalles que fue necesario resolver, para el mejor funcionamiento del proyecto en general.

El programa; primero fue probado solo, esto se logró conectando juntos los pines de TXD y RXD pertenecientes al puerto serie de la computadora y de esta forma lo que mandaba la PC era lo mismo que se recibia, comprobando así que el programa funcionaba satisfactoriamente al recibir los dos bytes que eran mandados, y con el mismo valor.

Para poder llegar a las pruebas finales de la interfase se requirió algún tiempo, ya que la parte de control tomo tiempo ajustarla adecuadamente a nuestras necesidades, esto nos orillo a crear variables artificiales, si así podemos llamarles; es decir, variables externas creadas y controladas por nosotros para facilitar el paso de un estado a otro. La necesidad de estas variables se basa en que las banderas de los UARTS que nosotros utilizamos para el control y que creíamos se daban en un determinado tiempo no era así, creandonos problemas graves de control.

La prueba principal, entre muchas, que nos permitio estar seguros del buen funcionamiento de la interfase; fué la que se realizó para comprobar que la interfase recibia dos bytes y estos eran llevadas en paralelo a la tarjeta de forma correcta, comprobando así que las velocidades entre la interfase y la computadora concuerdan; el siguiente paso fue ver si la interfase mandaba en serie los dos bytes que la tarjeta regresa al ser alimentada con datos, pero para esto fué necesario colocar un ciclo en la computadora para que nunca saliera de transmitir los mismos datos y de esta forma tener siempre los mismos datos de salida para poderlos ver en el osciloscopio; de esta manera pudimos comprobar que la interfase mandaba dos bytes a la computadora y la velocidad era correcta.

Se realizarón unas pruebas juntando la interfase, la tarjeta y un programa muy simple, el cual solo mandaba dos datos y recibía los datos que mandaba la interfase; este programa no tenía ventanas ni ningún tipo de opciones que el programa final contiene. Al hacer estas pruebas se encontró que la computadora siempre recibía dos datos como era requerido, pero el problema era que siempre fueron los mismos, así que se procedio a revisar si la tarjeta estaba respondiendo ya que estabamos seguros que la PC estaba enviando bien los datos de prueba y la interfase recibiendolos correctamente; ya que los datos enviados por la PC eran encontrados a la entrada de la tarjeta. Así que el problema se encontraba en la respuesta de la tarjeta, lo cual fue comprobado. Los opto-acopladores que tiene la tarjeta en cada una de sus entradas no eran alcanzados a ser activados con el voltaje que provenía de los buffers conectados a las salidas en paralelo de los UARTS; este voltaje era de alrededor de 4 Volts, así que se tuvo que amplificar las salidas en paralelo de los UARTS para poder hacer trabajar a la tarjeta.

Resuelto el problema de la tarjeta se procedió a verificar si los dos bytes recibidos por la PC eran correctos, y así fue. El siguiente paso consistio en desarrollar mas software, para tener opción de checar la tarjeta por completo, o linea por linea y tener una ayuda para algún error, además de indicarle al usuario a través de ventanas los pasos a seguir para la utilización del proyecto; quedando así un programa largo y lleno de ventanas.

Sin embargo al probar juntos la interfase, la tarjeta y el nuevo programa se encontró que la computadora solo leia uno de los bytes perdiendo el otro. Al principio se pensó que se perdía el último byte que mandaba la interfase a la PC, así que se trató de agilizar el programa para que en cuanto leyera el primer byte inmediatamente leyera el segundo, pero no se logró ningún resultado positivo. Se continuaron las pruebas, esta vez se trató de verificar si era cierto que el byte que era perdido era el segundo, así que se estuvo cambiando los bytes de entrada a la interfase para ver como cambiaba su respuesta el único byte que se recibía en la interfase, llegando a la conclusión de que no era el segundo byte que la interfase manda a la PC, sino era el primero. Este problema trató de solucionarse a través de software, provocando retardos en el programa antes de que fuera a traer el primer dato del puerto de entrada, este intento no fue fructoso así que decidimos mandar dos veces el primer dato que manda la

interfase a la PC para asegurar que lea ese dato la segunda vez que es enviado, y dio resultado. Para lograr mandar dos veces se tuvo que modificar un poco la parte de control de la interfase, aumentando un contador el cual nos da la pauta para continuar al segundo byte-resultado.

Consideramos que el uso de ventanas y letreros de error provoca retardos al programa los cuales a su vez provocarón la perdida del primer byte.

Por último se procedió a probar de nuevo llegando a la conclusión de que tanto el programa como la interfase trabajan correctamente juntos, lo cual algunas veces parecía que no era así pero esto es debido a la tarjeta con la cual se hacen la pruebas no esta trabajando del todo bien y en algunas ocasiones no responde como es debido a los bytes de prueba.

146498

C O N C L U S I O N E S

A través del desarrollo de este proyecto hemos podido concluir que el ultimar los detalles es aún mas pesado que el diseño mismo del proyecto, y que en el caso de localizar un error grave en la base del proyecto, lo mas sano es truncar su desarrollo y regresar a atacar el problema de raíz, ya que continuar de esta forma así con una base erronea acarreará mas problemas y a la larga llevará mas tiempo resolverlos.

Tambien pudimos llegar a la conclusion de que las velocidades en una comunicación asíncrona son criticas, ya que estamos haciendo uso de relojes independientes y estos deben de tener una cierta exactitud para poder asegurar la transmisión y recepción correcta. Ademas de que tanto una parte como otra; transmisor y receptor ambos deben estar programados de la misma forma para poder tener datos confiables. Es decir, que ambos manden o reciban datos con la misma paridad o sin ella, así como con un bit de comienzo y uno o dos bits de parada según como se requiera.

Tambien pudimos concluir que antes de comenzar el diseño de un proyecto hay que estar completamente seguros de lo que necesitamos y de que erramientas disponemos para atacar el problema, el cual debe de estar tambien completamente definido. De no contar con objetivos de trabajo claros y con una idea precisa del problema se corre el peligro de entrar en un proyecto largo, canzado y en algunas ocasiones parecera que sin solución.

B I B L I O G R A F I A

Notas sobre los UARTS.

Manuales de INTEL.

Manual de TTL.

Manual de Linear.

Manual del usuario de la PC IBM.

Manual de ECG.

APENDICE A

UARTS:

Receptor-Transmisor Universal Asíncrono (UART).

Los Receptores-Transmisores Universales Asíncronos o UART han sido desarrollados de forma tal que todos han sido incorporados a un circuito integrado standard de 40 pines.

Hay algunas funciones en el ship que son programadas por el usuario de forma tal que formatos de datos seriales-asíncronos pueden ser acomodados.

Los UARTS, son fabricados por diferentes casa, entre ellas la Texas Instrument que fabrica el TM56012; la Western Digital Corporation con el TR1602 y la General Instrument Corporation con el AY-3-1015, aunque los números de pines y las descripciones funcionales serán las mismas para dispositivos compatibles.

Los UART se componen basicamente de 3 partes, una de control, otra de recepción y una última de transmisión. Las señales de control del UART son aquellas que son comunes a ambas secciones, la de transmisión y la de recepción. Estas incluyen el Master Reset, pin 21, y otras 6 señales de control que son descritas mas adelante. Estas 6 señales controlan el formato del dato que está siendo transmitido y recibido. El formato para la sección de Recepción y de Transmisión para cada chip es el mismo, dado que las mismas 6 lineas son usadas para programarlas a ambas.

TABLAS DE FUNCIONES DEL UART

CONTROL

Num. de Pin	Nombre
21	MR -> Master Reset
34	CS -> Control Strobe
35	NP -> No Parity
39	PE -> Parity or Parity Even
36	SB -> Stop Bits
37	NB2-> Number of
38	NB1-> Data bits
16	SWE-> Status Words Enable

TRANSMISION

Num. de Pin	Nombre
26	DB1-> Data Bit 1
27	DB2-> Data Bit 2
28	DB3-> Data Bit 3
29	DB4-> Data Bit 4
30	DB5-> Data Bit 5
31	DB6-> Data Bit 6
32	DB7-> Data Bit 7
33	DB8-> Data Bit 8
40	Clock
25	SO -> Serial Output
22	TMBT->Transmiter Buffer Empty Flag.
23	DS -> Data Strobe
24	EOC-> End of Character Flag

RECEPCION

Num. de Pin	Nombre
12	RD1-> Data Bit 1
11	RD2-> Data Bit 2
10	RD3-> Data Bit 3
9	RD4-> Data Bit 4
8	RD5-> Data Bit 5
7	RD6-> Data Bit 6
6	RD7-> Data Bit 7
5	RD8-> Data Bit 8
17	Clock
20	SI -> Serial Input
4	RDE-> Received Data Enable
18	DRR-> Data Ready Reset
19	RDA-> Received Data Available
13	PE -> Parity Error
14	FE -> Framing Error
15	OR -> Overrun Error

EXPLICACION DE LAS TABLAS:

CONTROL:

Master Resert, pin 21: Un 1 lógico en esta entrada da un Reset al UART. Esta entrada debe de estar en un estado 0 lógico para operaciones normales. El UART debe de ser reseteado cada vez que un voltaje de encendido es aplicado a él.

No Parity, pin 35: Un 1 lógico en esta entrada eliminará la presencia del bit de paridad en los bits transmitidos. Por lo cual, el receptor no esperará recibir un bit de paridad entre los bits. Si este bit es un cero lógico, el bit de paridad, par ó impar, será colocado en la transmisión en serie de los bits entre el último bit del dato (MSB) y el bit de parada. Por lo tanto, el receptor esperará recibir un bit de paridad si el pin 35 es un cero lógico (en nuestra interfase su utiliza un 0 logico).

Parity Select, pin 39: Un 1 lógico en esta entrada selecciona el modo de paridad par y un cero lógico selecciona el modo de paridad impar. El modo de selección no es usado por la lógica interna del UART si el pin 35 (No parity input) está en el estado uno lógico.

Number of Stop bits, pin 36: Un cero lógico en esta entrada seleccionará un bit de parada, mientras que un uno lógico seleccionará dos bits de parada.

Number of Data bits, pins 37 y 38: Estos pines permiten la selección de cinco, seis, siete u ocho bits de datos en serie, entre el bit de comienzo y el bit de parada. Si cinco bits son seleccionados, entonces habrá cinco bits de datos en la información transmitida y la sección de recepción del UART esperará recibir cinco bits de datos.

Estas dos entradas son programadas como sigue:

NB-2(pin 37)	NB-1(pin 38)	Data Bits
0	0	5
0	1	6
1	0	7
1	1	8

Control Strobe, pin 34: Un uno lógico en esta entrada al UART permitirá la entrada de los cinco bits de control dentro de la circuiteria de respectiva del UART. Esta entrada deberá recibir un pulso de un uno lógico, o deberá estar permanentemente en un uno lógico, como usualmente se hace en nuestra interfase.

TRANSMISION:

Clock, pin 40: La sección de transmisión del UART acepta un set en paralelo mayor a 8 bits de datos, los cuales son formateados por los bits de control, y despues los mandanda en forma serial. Para hacer esto, primero se requiere una señal de reloj. En la mayoría de las aplicaciones de los UART, la señal de reloj debe ser 16 veces la de el data rate.

Data Strobe, pin 23: La sección de transmisión del UART también usa tres señales de control para manejar la transmisión actual de información en serie. El Data Strobe es una señal que carga la palabra de 8 bits dentro del buffer de transmisión. Un cero lógico en el Data Strobe causa que se realize esta acción. Si el transmisor está todavía mandando el dato, la transmisión es completada antes de que el nuevo dato sea transmitido desde el buffer de transmisión al registro del transmisor, desde el cual es transmitido.

Transmiter Buffer Empty, pin 22: Una señal o bandera de que el buffer de transmisión está vacío (TMBT) está provista como una salida que indica que el buffer de transmisión está vacío (uno lógico) o que éste todavía contiene la palabra que tiene que ser aún transmitida (cero lógico).

Cuando una palabra ha sido transferida desde el buffer de transmisión al registro de transmisión para ser mandada, la bandera del buffer se va a uno lógico, indicando que la próxima palabra puede ser recibida por la sección del transmisor del UART.

End of Character, pin 24: Una bandera adicional está provista para indicar si está o no la transmisión en proceso. La bandera de fin de caracter (EOC) tiene un uno lógico cuando no se está transmitiendo y un cero lógico cuando se está transmitiendo. El caracter está compuesto de la palabra de datos de 5 a 8 bits, y los bits de control.

146498

Dado que la sección de transmisión está compuesta de dos registros, está puede estar transmitiendo un caracter mientras otro está esperando.

Serial Output, pin 25: La salida serial (SO) del transmisor está disponible en el pin 25. La salida serial es una representación de bit compatible con el TTL.

RECEPCION:

Clock, pin 17: La sección de recepción del UART es de alguna manera más complicada que la sección de transmisión, dado que tenemos bits de comienzo, bits de parada, bits de paridad y los bits de los datos deben de ser reconocidos y ejecutados.La sección de recepción también requiere una señal de clock que es 16 veces la velocidad con la que se va ha recibir el bit. La señal de clock es recibida y aplicada en el pin 17.

Received Data Enable, pin 4: Una vez que un paquete de bits en serie ha sido recibido, este es transferido en paralelo desde el registro de recepción hacia el buffer en la sección de recepción de UART. Las ocho salidas de esta sección pueden tener tres estados; uno lógico, cero lógico y tercer estado. Este último es un estado de alta impedancia que desconecta las salidas del receptor del UART de los dispositivos a los cuales está conectado. Esta entrada debe tener un cero lógico para permitir la recepción de datos, si tiene un uno lógico los ocho datos de entrada estarán en estado de alta impedancia, desconectándolos del sistema.

Data Ready Reset, pin 18; y Received Data Available, pin 19: La habilitación de los datos recibidos es indicada por el estado de la bandera del dato recibido y disponible, pin 19. Un uno lógico en esta salida indica que un set completo de bits de datos ha sido recibido y transferido al buffer receptor. Un cero lógico indica que un set de bits de datos no ha sido recibido aún. La bandera Data Ready Reset (DRR) pin 18, está disponible como una entrada por lo tanto la bandera Received Data Available debe ser reseteada despues de que la bandera ha sido censada y que el dato ha sido usado por el sistema de recepción. Esta acción activa la bandera Received Data Available, por lo tanto debe ser usada para sensar y habilitar el próximo set de bits de datos que serán enviados serialmente a la sección de recepción del UART. Un cero lógico aplicado a la entrada Data Ready Reset reseteará la bandera Received Data Available.

Parity Error,pin 13: Esta salida se va a uno lógico si la paridad de la palabra de datos recibida no concuerda con la paridad que ha sido seleccionada para el UART en el pin 39, Parity Select. La salida de Parity Error no tiene significado si no se ha seleccionado un tipo de paridad en cuyo caso el pin 35 estará en uno lógico.

Framing Error, pin 14: Un uno lógico en esta salida es usado para indicar que el receptor no ha detectado un bit de parada, o dos bits de parada, en el paquete de bits recibido.

Overrun, pin 15: Un uno lógico en esta salida indica que la bandera Received Data Available no ha sido reseteada despues de que el caracter previo ha sido recibido. Por lo tanto, el caracter que se acaba de recibir ha sido escrito sobre su antecesor. Por lo tanto la circuiteria externa al UART debe resetear la bandera Received Data Available después de que cada nueva palabra de datos es censada y utilizada en el UART.

Status Word Enable, pin 16: Parity Error, Overrun Error y Framing Error, así como Transmitter Buffer Empty y Received Data Available, tienen tres estados de salida. Estas salidas son controladas simultaneamente por la entrada Status Word Enable, pin 16. Cuando esta entrada está en cero lógico, la salida de las 5 banderas están activas (uno o cero lógico), pero si Status Word Enable está en uno lógico, las cinco banderas de salida están en alta impedancia o desconectadas.

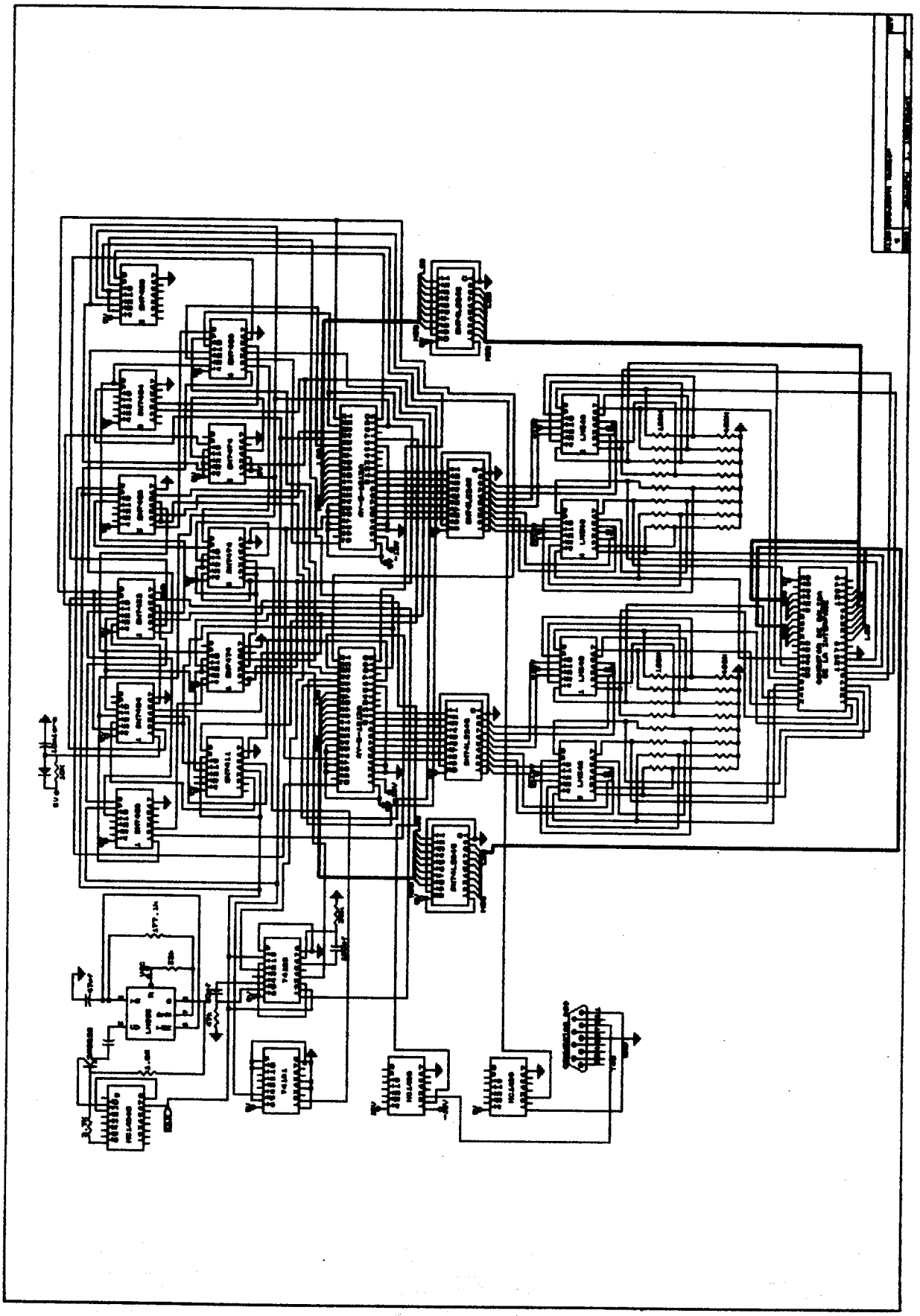
APENDICE B

MAPA DE REGISTROS

Registros	7	6	5	4	3	2	1	0
TxD	Tx Data Bit 7	Tx Data Bit 6	Tx Data Bit 5	Tx Data Bit 4	Tx Data Bit 3	Tx Data Bit 2	Tx Data Bit 1	Tx Data Bit 0
RxD	Rx Data Bit 7	Rx Data Bit 6	Rx Data Bit 5	Rx Data Bit 4	Rx Data Bit 3	Rx Data Bit 2	Rx Data Bit 1	Rx Data Bit 0
BAL	BRGA LSB Divide Count (DLAB = 1)							
BAH	BRGA MSB Divide Count (DLAB= 1)							
IER	0	0	0		Modem Interrupt Enable	Rx Machine Interrupt Enable	Tx Data Interrupt Enable	Rx Data Interrupt Enable
IIR	0	0	0	0	0	Active Interrupt Bit 1	Active Interrupt Bit 0	Interrupt Pending
LCR	DLAB Divisor Latch	Set Break	Parity Mode Bit 2	Parity Mode Bit 1	Parity Mode Bit 0	Stop Bit Lenght Bit 0	Character Lenght Bit 1	Character Lenght Bit 0
MCR	0	0	0	Loopback Control Bit	OUT 2 Complement	0	RTS Complement	DTR Complement
LSR	0	TxM Status	TxD Empty	Break Detected	Framing Error	Parity Error	Overrun Error	Rx Data Available
MSR	DCD Input Inverted	RI Input Inverted	DSR Input Inverted	CTS Input Inverted	State Change in DCD	State (H-L) Change in RI	State Change in DSR	State Change in CTS

APENDICE C

146498



APENDICE D

```
eta.obj (twindow.h, keys.h)
int.obj
ry.obj (twindow.h, keys.h)
lp.obj (twindow.h, keys.h)
ndow.obj (twindow.h, keys.h)
pc.obj
```

```
/*****************************************************************************
*                                                                            **
*                                                                            **
*                          PROYECTO FINAL II                                 **
*                                                                            **
*                                                                            **
*                                                                            **
*                  Comunicar una computadoras atraves del                    **
*                              PUERTO SERIE                                  **
*                  con una interface para checar una tarjeta.                **
*                                                                            **
*                 ALUMNOS:                                                   **
*                          Acosta Gastelum Luz Fca.                          **
*                          Pulido Baños Alejandro.                           **
*                                                                            **
*****************************************************************************/

nclude <stdio.h>
nclude <dos.h>
nclude <conio.h>
nclude <alloc.h>
nclude <string.h>
nclude <stdlib.h>
nclude "twindow.h"
nclude "keys.h"
nclude "vecint.h"
nclude "prototi.h"

efine RII              0x3fa
efine TRUE               1
efine FALSE              0
efine num_baud_rate     32     /* doble de transmisiones posibles */
efine BASEINT            8     /* numero base para las interrupciones */
efine UART_COM1_INT      4     /* COM1 = IR@4 del registro de
                                  enmascaramiento del 8259 */
efine MASK_REGISTER   0x21     /* direccion del registro de
                                  enmascaramiento del 8259 */
efine Max                2     /* maximo numero de elementos de la cola*/
nsigned short int var1, var2, var3, var4, var5, var6;

oid checa_tarjeta(void);
oid inicializa(void);
oid proyecve(void);

                               /* da el menu de opciones */
har *opciones [] = {
                    " 1: Reset para la interfase ",
                    " 2: Checar el estado de la tarjeta ",
                    " 3: Correr Diagnostico por Bit",
                    " 4: Salir del Programa ",0
                      };
```

```
                int ini,fin; /* limites de la cola */
                unsigned short int info[Max];
                                /* informacion en la cola */
            } LISTA;

STA *colatran,*colares;         /* colas usadas */

te tabla_div[num_baud_rate]={9,0, 6,0,  4,23, 3,89,  3,0, 1,128,
                             0,192,  0,96,  0,64,  0,58,  0,48,
                             0,32, 0,24, 0,16,  0,12,  0,6 };

signed short int arbit[32]={1,0, 2,0, 4,0, 8,0, 16,0, 32,0, 64,0, 128,0,
                            0,1, 0,2, 0,4, 0,8, 0,16, 0,32, 0,64, 0,128 };

t lowbyte;                      /* baud rate dividor parte baja */
t highbyte;                     /* baud rate parte alta */
t port=0x3f8;                   /* se utiliza COM1 */
ndtype vector;                  /* en este vector se almacena el vector
                                   actual que se encuentra en la posicion
                                   reservada para las interrupciones
                                   producidas por COM1 */
ndtype vector1, vector2;
t trans_parada=TRUE;            /* trans_parada es activa mientras no halla */
                                /* datos en la cola */
t bandera=FALSE;                /* bandera para checar que la tarjeta esta */
                                /* o no en buenas condiciones */
t band;
signed short int colaresp[2];/* arreglo para comparar con la cola de recepcior
signed short int arre[2];
signed short int arre1[2];
t conta=0;
t cuenta=0;
t cuent,cuent2;
t con=0;
NDOW  *wndA;                    /* ventana de presentacion */
NDOW  *wndX;

id set_baud_rate()
                                /* selecciona la frecuencia de transmision */
                                /* que es de 2400 baude rate*/

int  in,i;

in = 'E';
i = (in - 'A')*2;
highbyte = tabla_div[i];
lowbyte = tabla_div[i+1];


t LLENA(LISTA *cola)
                                /* se verifica si la cola esta llena */

nt full;

f((cola->fin+1)%(Max+1)==cola->ini)
 full=TRUE;
lse
 full=FALSE;
eturn(full);


t VACIA(LISTA *cola)
                                /* se verifica si la cola esta vacia */

nt empty;
```

```
f (cola->ini==cola->fin)
 empty=TRUE;
lse
 empty=FALSE;
eturn(empty);


id ENCOLA(LISTA *cola, unsigned short int elemento)
                                /* encola los elementos que se transmiten */
                                /* y se reciben */

f (LLENA(cola)==TRUE)
  printf("\n I'm sorry. The list is full \n");
lse
  {
   cola->info[cola->fin]=elemento;
   cola->fin=((cola->fin+1)%(Max+1));
  }


pid DESENCOLA(LISTA *cola,unsigned short int *elemento)
                                /* desencola los elementos que se transmiten*/
                                /* y se reciben */

if (VACIA(cola) != TRUE)
  {
   *elemento=cola->info[cola->ini];
   cola->ini=((cola->ini+1)%(Max+1));
  }


oid TRANSMITIR()
                                /* En este procedimiento se encolan los */
                                /* datos a transmitir en colatran */

unsigned short int elemento;

 if (VACIA(colatran) != TRUE)
  {
    DESENCOLA(colatran,&elemento);
    outportb(port,elemento);
   }
  else
    trans_parada=TRUE;
}

void RECIBIR()
                                /* En este procedimiento se encolan los */
                                /* datos que se reciben en la cola de */
                                /* recepcion */
{
 if (LLENA(colares) == FALSE )
    ENCOLA(colares,inportb(port));
    wprintf(wndX,"%i",inportb(0x3fd));
    wprintf(wndX,"%i",inportb(port));
}

void interrupt PTO_SERIE()
                                /* Este procedimiento analiza el estado de */
                                /* las interrupciones en el puerto serie   */
{
 int interrupcion;

 while (((interrupcion=inportb(RII))&1)==0)
```

```
switch((interrupcion&7)>>1)
 {
  case 1:
          wprintf(wndX,"%i",(interrupcion&7)>>1);
          TRANSMITIR();
          break;
  case 2:
          wprintf(wndX,"%i",(interrupcion&7)>>1);
          RECIBIR();
          break;
 }
 outportb(0x20,0x20);
}


oid interrupt DEMON()

unsigned short int elemento;

if (trans_parada==TRUE)
{
    if (VACIA(colatran) == FALSE)
    {
     DESENCOLA(colatran,&elemento);
     outportb(port,elemento);
     trans_parada=FALSE;
     }
  }
  GenInt(CLKINTBK);
}

ys_on()
                                /* inicializa al 8250 */
 {
  outportb(port+3,0x80);        /* se elije al latch divisor del
                                   registro de control de linea */
  outportb(port,lowbyte);       /* se escribe la parte baja del latch divisor */
  outportb(port+1,highbyte);    /* se escribe la parte alta del latch divisor */
  outportb(port+3,0x03);        /* se envian 8 bits de datos y uno de stop */
  outportb(port+4,0x0B);        /* se programa el modem para recibir datos */
  outportb(port+1,0x03);        /* se habilitan las interrupciones
                                   para recepcion de datos en el 8250 */
  disable();                    /* se deshabilitan las interrupciones */
  vector = getvect(BASEINT+UART_COM1_INT);
                                /* se obtiene el vector actual y se almacena */
                                /* para cuando el sistema sea regresado a */
                                /* sus condiciones iniciales */
  setvect(BASEINT+UART_COM1_INT,PTO_SERIE);
                                /* se coloca el nuevo vector */
  outportb(MASK_REGISTER,inportb(MASK_REGISTER) & ~(1<<4));
                                /* se habilitan las interrupciones */
                                /* correspondientes a la linea IRQ4 del */
                                /* controlador de interrupciones (8259) */
  vector1 = getvect(BASEINT+CLKINT);
  vector2 = getvect(USRCLKINT);
  setvect(CLKINTBK, vector1);
  setvect(BASEINT+CLKINT,DEMON);
                                /* activa el demonio en el reloj */
  enable();                     /* se habilitan las interrupciones */
}

sys_off()
                                /* regresa al sistema a sus condiciones inicial
 {
  disable();                    /* se deshabilitan las interrupciones */
```

```
  outportb(port+1,0x00);          /* se deshabilitan las interrupciones del UART */
                                  /* se deshabilitan las interrupciones */
                                  /* correspondientes a la linea IR04 del */
                                  /* controlador de interrupciones (8259) */
  setvect(BASEINT+UART_COM1_INT,vector);
                                  /* se coloca el viejo vector */
  setvect(BASEINT+CLKINT,vector1);
  setvect(USRCLKINT,vector2);
  enable();                       /* se habilitan las interrupciones */
 }

oid inicializa()
                                  /*se dan las condiciones de comienzo del program

 colares=(LISTA*)malloc(sizeof(LISTA));
 colatran=(LISTA*)malloc(sizeof(LISTA));
 colaresp[0]=255;
 colaresp[1]=255;
 colaresp[2]=0;
 colaresp[3]=0;
 arre[0]=0;arre[1]=0;
 arre1[0]=255;arre1[1]=255;


oid compara(void)
                                  /* se compara lo recibido con lo esperado */

 WINDOW *wndC;                    /* ventana despliega resultado */
 WINDOW *wndD;                    /* ventana despliega mensaje */
 WINDOW *ewnd;                    /* ventana mensaje de error */
 char sal='';

 colares->ini=0;
 bandera=FALSE;
 band=TRUE;
 wndC = establish_window(25,22,3,40);
 set_title(wndC,"Resultado");
 set_colors(wndC, ALL, RED, YELLOW, BRIGHT);
 display_window(wndC);
 do
  {
   if ((int)colares->info[colares->ini] != colaresp[con])
   {
    wprintf(wndC,"Lo siento, la tarjeta se encuentra mal");
    bandera=TRUE;
   }
   else
   {
    colares->ini=((colares->ini+1)%(Max+1));
    con++;
   }
   if (((con==2)||(con==4))&&(bandera==FALSE))
   {
     bandera=TRUE;
     band=FALSE;
    }
   }while(bandera==FALSE);
  if (band==FALSE)
  {
   wprintf(wndC,"La tarjeta se encuetra en buen estado!");
   }
 wndD = establish_window(45,1,3,31);
 set_colors(wndD, ALL, BLUE, YELLOW, BRIGHT);
 display_window(wndD);
 wprintf(wndD,"Oprime <ENTER> para continuar");
```

```
  {
   sal=get_char();
   if (sal!='\r')
   {
    ewnd = establish_window(2,1,3,25);
    set_colors(ewnd, ALL, RED, YELLOW, BRIGHT);
    set_title(ewnd, " ERROR! ");
    display_window(ewnd);
    wprintf(ewnd,"Te equivocaste de tecla");
    putchar(BELL);
    delay(300);
    delete_window(ewnd);
    }
  }while (sal!='\r');
 delete_window(wndD);
 delete_window(wndC);
 delete_window(wndA);
 delete_window(wndX);


oid comparar(void)

 WINDOW *wndC;                    /* ventana despliega resultado */
 WINDOW *wndD;                    /* ventana despliega mensaje */
 WINDOW *ewnd;                    /* ventana mensaje de error */
 unsigned short int elemento=0;
 int i,residuo;
 int buf[8];
 char sal='';
 char ret='';

 colares->ini=0;
 bandera=FALSE;
 wndC = establish_window(25,22,3,51);
 set_title(wndC,"Resultado");
 set_colors(wndC, ALL, RED, YELLOW, BRIGHT);
 display_window(wndC);
 if (conta>=8)
 {
  elemento = (int)colares->info[colares->ini];
  for(i=0; i<8; i++)
   {
    residuo = elemento % 2;
    if (residuo==1)
      buf[i] = 1;
    else
      buf[i] = 0;
    elemento = elemento/2;
    }
  if (buf[conta-8]==0)
  {
   wprintf(wndC," El diagnostico para el bit ");
   wprintf(wndC,"%d",conta);
   wprintf(wndC," fue el correcto! ");
   }
  else
  {
   wprintf(wndC,"   Lo siento, el resultado del bit ");
   wprintf(wndC,"%d",conta);
   wprintf(wndC," esta mal");
  }
 }
 else
 {
  elemento = (int)colares->info[colares->ini+1];
```

```
   {
    residuo = elemento % 2;
    if (residuo==1)
      buf[i] = 1;
    else
      buf[i] = 0;
    elemento = elemento/2;
    }
 if (buf[conta]==0)
 {
  wprintf(wndC," El diagnostico para el bit ");
  wprintf(wndC,"%d",conta);
  wprintf(wndC," fue el correcto! ");
  }
 else
 {
  wprintf(wndC,"   Lo siento, el resultado del bit ");
  wprintf(wndC,"%d",conta);
  wprintf(wndC," esta mal");
 }
}
wndD = establish_window(45,1,3,31);
set_colors(wndD, ALL, BLUE, YELLOW, BRIGHT);
display_window(wndD);
wprintf(wndD,"Oprime <ENTER> para continuar");
do
{
 sal=get_char();
 if (sal!='\r')
 {
  ewnd = establish_window(2,1,3,25);
  set_colors(ewnd, ALL, RED, YELLOW, BRIGHT);
  set_title(ewnd, " ERROR! ");
  display_window(ewnd);
  wprintf(ewnd,"Te equivocaste de tecla");
  putchar(BELL);
  delay(300);
  delete_window(ewnd);
  }
}while (sal!='\r');
delete_window(wndD);
delete_window(wndC);
if (conta < 15)
{
 wprompt(wndA,0,11,"-------------------------------------------------");
 wprompt(wndA,5,12,"Quieres regresar al Menu Anterior 'S/N'");
 cursor(0,25);
 do
 {
  ret = get_char();
  if ((ret!='s')&&(ret!='S')&&(ret!='n')&&(ret!='N'))
   {
    error_message("Te equivocaste de Tecla");
    delay(300);
    clear_message();
    }
  }while ((ret!='s')&&(ret!='S')&&(ret!='n')&&(ret!='N'));
 if ((ret=='S')||(ret=='s'))
 {
  conta = 17;
  }
}
delete_window(wndA);
delete_window(wndX);
}
```

```
d diagnos(void)

INDOW *wndC,*ewnd;
nsigned short int elemento=0;
nt i,residuo,contador,cont;
nt buf[8];
nar sal='';

>

ndX = establish_window(0,0,3,20);
set_colors(wndX, ALL, BLUE, YELLOW, BRIGHT);
set_title(wndX,"Banderas");
display_window(wndX);
wndA = establish_window(10, 5, 15, 50);
set_border(wndA,3);
set_colors(wndA, ALL, BLUE, YELLOW, BRIGHT);
set_title(wndA,"Corriendo Diagnostico");
display_window(wndA);
contador=0;
cont=1;
wprompt(wndA,3,1,"Estoy corriendo diagnostico para ");
wprintf(wndA,"bit ==> ");
wprintf(wndA,"%d",conta);
wprompt(wndA,5,4,"Estoy Transmitiendo...");
cursor(0,25);
wndC = establish_window(25,22,3,59);
set_title(wndC,"Mensaje");
set_colors(wndC, ALL, BLUE, WHITE, BRIGHT);
display_window(wndC);
wprintf(wndC,"Presiona dos veces 't' para mandar los datos a la tarjeta");
sys_on();
while (contador<2)
{
 do
 {
  sal = get_char();
  if (sal!='t')
  {
   ewnd = establish_window(51,1,3,25);
   set_colors(ewnd, ALL, RED, YELLOW, BRIGHT);
   set_title(ewnd, " ERROR! ");
   display_window(ewnd);
   wprintf(ewnd,"Te equivocaste de tecla");
   putchar(BELL);
   delay(300);
   delete_window(ewnd);
   }
  }while (sal!='t');
 if (LLENA(colatran)==FALSE)
 {
  wprintf(wndA,"\n                              ");
  elemento=arbiticuenta];
  ENCOLA(colatran,elemento);
  for(i=0; i<8; i++)
  {
   residuo = elemento % 2;
   if (residuo==1)
     buf[i] = 1;
   else
     buf[i] = 0;
   elemento = elemento/2;
   }
  for(i=7; i>=0; i--)
    wprintf(wndA,"%i",buf[i]);
```

```
    cuenta++;
    }
   contador++;
   }
  delete_window(wndC);
  wprompt(wndA,5,8,"Estoy Recibiendo...");
  cursor(0,25);
  while ((VACIA(colares) == FALSE)&&(cont<3))
  {
   DESENCOLA(colares,&elemento);
   for(i=0; i<8; i++)
   {
    residuo = elemento % 2;
    if (residuo==1)
      buf[i] = 1;
    else
      buf[i] = 0;
    elemento = elemento/2;
    }
   wprintf(wndA,"%i",cont);
   wprintf(wndA," -> ");
   for(i=7; i>=0; i--)
   wprintf(wndA,"%i",buf[i]);
   wprintf(wndA,"\n                                ");
   cont++;
  }
  sys_off();
  comparar();
  conta++;
  colatran->ini=colares->ini=0;
  colatran->fin=colares->fin=0;
  cuent=1;
  cuent2=1;
  }while (conta<16);
}

void reset_in(void)
{
 unsigned short int elemento=0;
 int contador=0,cont=1;
 char sal='';

 wndX = establish_window(0,0,3,20);
 set_colors(wndX, ALL, BLUE, YELLOW, BRIGHT);
 set_title(wndX,"Banderas");
 display_window(wndX);
 wndA = establish_window(15, 8, 10, 50);
 set_border(wndA,3);
 set_colors(wndA, ALL, BLUE, YELLOW, BRIGHT);
 set_title(wndA,"Reset de la Interfase");
 display_window(wndA);
 wprompt(wndA,17,3,"Estoy Pensando...:");
 wprompt(wndA,0,6,"--------------------------------------------------");
 wprompt(wndA,4,7,"Presiona dos veces <ENTER> para continuar");
 cursor(0,25);
 while (contador<2)
 {
  do
   {
    sal = get_char();
    if (sal!='\r')
    {
     error_message("Te equivocaste de Tecla");
     delay(300);
     clear_message();
```

```
  }while (sal!='\r');
  if (LLENA(colatran)==FALSE)
  {
   elemento=arre[contador];
   ENCOLA(colatran,elemento);
   }
  contador++;
}
ursor(0,25);
hile ((VACIA(colares) == FALSE)&&(cont<3))

ESENCOLA(colares,&elemento);
ont++;
}
elete_window(wndA);
elete_window(wndX);


id proyecve()

INDOW *wndB;                    /* ventana del menu de opciones */
har **cp;
nt s = 0;

ys_off();
*delay(2000);*/
lrscr();
*cursor(0,25);
ndA = establish_window(0, 1, 23, 85);
et_border(wndA,4);
et_colors(wndA, ALL, RED, YELLOW+128, BRIGHT);
isplay_window(wndA);
prompt(wndA, 22, 2, "UNIVERSIDAD AUTONOMA METROPOLITANA");
prompt(wndA, 31, 5, "UNIDAD IZTAPALAPA");
prompt(wndA, 31, 9, "PROYECTO TERMINAL");
prompt(wndA, 29, 13,"CHECADOR DE TARJETAS");
prompt(wndA, 33, 17,"MARZO DE 1990");
ursor(1,26);
elay(4000);
elete_window(wndA);*/
ursor(0,25);
ndB = establish_window(0, 0, 6, 45);
et_title(wndB, "Menu");
et_colors(wndB, ALL, BLUE, GREEN, BRIGHT);
et_colors(wndB, ACCENT, GREEN, WHITE+128, BRIGHT);
isplay_window(wndB);
p = opciones;
hile (*cp)
 wprintf(wndB,"\n%s",*cp++);
hile (1) {
 colatran->ini=colares->ini=0;
 colatran->fin=colares->fin=0;
 cuent=1;
 cuent2=1;
 con=0;
 set_help("opciones", 40, 10);
 s = get_selection(wndB, s+1, "1234");
 if ((s == 0)||(s == 4))
 {
  clrscr();
  break;
  }
 if (s == FWD || s == BS)
 {
  s = 0;
```

```
 }
hide_window(wndB);
if ( s == 1 )
{
 sys_on();
 reset_in();
 sys_off();
 display_window(wndB);
 }
if ( s == 2 )
{
 checa_tarjeta();
 display_window(wndB);
 }
if ( s == 3 )
{
 diagnos();
 conta=0;
 cuenta=0;
 display_window(wndB);
 }

lose_all();


id checa_tarjeta(void)

INDOW *wndC,*ewnd;
nsigned short int elemento=0;
nt i,residuo,contador=0,cont=1;
nt buf[8];
har sal='';

o{
contador=0;
wndX = establish_window(0,0,3,20);
set_colors(wndX, ALL, BLUE, YELLOW, BRIGHT);
set_title(wndX,"Banderas");
display_window(wndX);
wndA = establish_window(10, 5, 15, 50);
set_border(wndA,3);
set_colors(wndA, ALL, BLUE, YELLOW, BRIGHT);
set_title(wndA,"Checando Tarjeta");
display_window(wndA);
wprompt(wndA,3,2,"Estos son los valores que la tarjeta debe de");
wprompt(wndA,7,3,"regresar cuando esta en buen estado:");
if (cuent==1)
 wprompt(wndA,16,4,"11111111 11111111");
else
 wprompt(wndA,16,4,"00000000 00000000");
wprompt(wndA,5,6,"Estoy Transmitiendo...");
cursor(0,25);
wndC = establish_window(25,22,3,59);
set_title(wndC,"Mensaje");
set_colors(wndC, ALL, BLUE, WHITE, BRIGHT);
display_window(wndC);
wprintf(wndC,"Presiona dos veces 't' para mandar los datos a la tarjeta");
sys_on();
while (contador<2)
{
 do
 {
  sal = get_char();
  if (sal!='t')
  {
```

```
    set_colors(ewnd, ALL, RED, YELLOW, BRIGHT);
    set_title(ewnd, " ERROR! ");
    display_window(ewnd);
    wprintf(ewnd,"Te equivocaste de tecla");
    putchar(BELL);
    delay(300);
    delete_window(ewnd);
    }
   }while (sal!='t');
  if (LLENA(colatran)==FALSE)
  {
   wprintf(wndA,"\n                                    ");
   if (cuent==1)
    elemento=arre[contador];
   else
    elemento=arre1[contador];
   ENCOLA(colatran,elemento);
   for(i=0; i<8; i++)
   {
    residuo = elemento % 2;
    if (residuo==1)
      buf[i] = 1;
    else
      buf[i] = 0;
    elemento = elemento/2;
    }
   for(i=7; i>=0; i--)
     wprintf(wndA,"%i",buf[i]);
   wprintf(wndA," ");
   }
  contador++;
  }
delete_window(wndC);
wprompt(wndA,5,10,"Estoy Recibiendo...");
cursor(0,25);
wndC = establish_window(45,22,3,19);
set_title(wndC,"Mensaje");
set_colors(wndC, ALL, RED, YELLOW, BRIGHT);
display_window(wndC);
wprintf(wndC,"Espera un momento");
cont=1;
while ((VACIA(colares) == FALSE)&&(cont<3))
{
 DESENCOLA(colares,&elemento);
 for(i=0; i<8; i++)
 {
  residuo = elemento % 2;
  if (residuo==1)
   buf[i] = 1;
  else
   buf[i] = 0;
  elemento = elemento/2;
  }
 wprintf(wndA,"%i",cont);
 wprintf(wndA," -> ");
 for(i=7; i>=0; i--)
  wprintf(wndA,"%i",buf[i]);
 wprintf(wndA,"\n                                    ");
 cont++;
 }
delete_window(wndC);
sys_off();
compara();
colatran->ini=colares->ini=0;
colatran->fin=colares->fin=0;
```

```
 }while (cuent<3);
}

oid main()
 {
  inicializa();                  /* condiciones iniciales de las colas */
  set_baud_rate();               /* seleccion de la velocidad de trabajo */
  sys_on();                      /* se cambian las condiciones de trabajo */
  proyecve();                    /* se corre el programa con ventanas */
 }
```